

Diego Gómez Cota A00824758
Mariana Martínez Celis González A01194953

Propuesta Sintaxis Lenguaje

El lenguaje PecanPy es un lenguaje básico orientado a objetos.

Lista de tokens:

program	less_than
id	equal_to
semicolon	not_equal_to
main	float
open_parenthesis	string
close_parenthesis	bool
open_key	function
close_key	returns
class	void
is	if
constructor	else
var	read
comma	write
group	for
assign	in
open_bracket	while
int	dot
close_bracket	at_class
and	return
or	obj
plus	int_value
minus	float_value
multiplication	string_value
division	float_value
greater_than	

Diagramas de sintaxis:

https://drive.google.com/file/d/1GKWj0RFp46UScyi1BNFTAyDA2f_G24-F/view?usp=sharing

Consideraciones semánticas básicas:

Jerarquía de Operadores (Mayor a menor)

Paréntesis ()
Multiplicación y División * /
Suma y Resta + -
Operadores relacionales > < == !=
Operadores Lógicos &&
Asignación =

Type Matching

	*	/	+	-	Operadores relacionales	Operadores Lógicos
int int	int	float	int	int	bool	bool
int float	float	float	float	float	bool	bool
int bool	err	err	err	err	err	bool
int group	err	err	err	err	err	err
int string	err	err	err	err	err	err
float float	float	float	float	float	bool	bool
float bool	err	err	err	err	err	bool
float group	err	err	err	err	err	err
float string	err	err	err	err	err	err
bool bool	err	err	err	err	bool (error para todos los que no sean == o !=)	bool
bool group	err	err	err	err	err	err
bool string	err	err	err	err	err	err
group	err	err	err	err	err	err

group						
group string	err	err	err	err	err	err
string string	err	err	err	err	err	err

Gramática formal

Nota: Los terminales están expresados en mayúsculas, mientras que los no terminales se expresan en minúscula.

program ->

- PROGRAM ID SEMICOLON declaration_loop MAIN OPEN_PARENTHESIS CLOSE_PARENTHESIS OPEN_KEY statement_loop CLOSE_KEY

declaration_loop ->

- declaration declaration_loop
- epsilon

statement_loop ->

- statement statement_loop1

statement_loop1 ->

- statement statement_loop1
- epsilon

declaration ->

- class_declaration
- variable_declaration
- function_declaration

variable ->

- ID variable1

variable1 ->

- OPEN_BRACKET hyper_exp CLOSE_BRACKET
- DOT ID
- epsilon

class_declaration ->

- CLASS ID class_declaration1 OPEN_KEY class_body CLOSE_KEY SEMICOLON constructor class_declaration2

class_declaration1 ->

- IS ID
- epsilon

class_declaration2 ->

- class_function class_declaration2
- epsilon

class_body ->

- class_body1 class_body3

class_body1 ->

- variable_declaration class_body2

class_body2 ->

- variable_declaration class_body2
- epsilon

class_body3 ->

- class_function_declaration class_body4

class_body4 ->

- class_function_declaration class_body4
- epsilon

constructor ->

- CONSTRUCTOR ID OPEN_PARENTHESIS parameter CLOSE_PARENTHESIS
OPEN_KEY statement_loop CLOSE_KEY

variable_declaration ->

- VAR data_type ID SEMICOLON
- GROUP ID ASSIGN data_type OPEN_BRACKET INT_VALUE CLOSE_BRACKET
SEMICOLON
- OBJ ID ASSIGN ID OPEN_PARENTHESIS variable_declaration1
CLOSE_PARENTHESIS SEMICOLON

variable_declaration1 ->

- hyper_exp_loop
- epsilon

statement ->

- assignment
- conditional
- cycle

- read
- write
- function_call
- variable_declaration

assignment ->

- variable ASSIGN hyper_exp SEMICOLON

hyper_exp ->

- super_exp hyper_exp1

hyper_exp1 ->

- AND super_exp
- OR super_exp
- epsilon

super_exp ->

- exp super_exp1

super_exp1 ->

- GREATER_THAN exp
- LESS_THAN exp
- EQUAL_TO exp
- NOT_EQUAL_TO exp
- epsilon

exp ->

- term exp1

exp1 ->

- PLUS term exp1
- MINUS term exp1
- epsilon

term ->

- factor term1

term1 ->

- MULTIPLICATION factor term1
- DIVISION factor term1
- epsilon

factor ->

- function_call
- FLOAT_VALUE
- INT_VALUE
- BOOL_VALUE
- STRING_VALUE
- variable
- OPEN_PARENTHESIS hyper_exp CLOSE_PARENTHESIS

data_type ->

- INT
- FLOAT
- STRING
- BOOL

class_function_declaration ->

- FUNCTION ID OPEN_PARENTHESIS parameter CLOSE_PARENTHESIS RETURNS
return_arg SEMICOLON

return_arg ->

- data_type
- VOID

parameter ->

- data_type ID parameter1
- epsilon

parameter1 ->

- COMMA data_type ID parameter1
- epsilon

conditional ->

- IF OPEN_PARENTHESIS hyper_exp CLOSE_PARENTHESIS OPEN_KEY
statement_loop CLOSE_KEY conditional1

conditional1 ->

- ELSE OPEN_KEY statement_loop CLOSE_KEY
- epsilon

cycle ->

- FOR OPEN_PARENTHESIS ID IN ID CLOSE_PARENTHESIS cycle1
- WHILE OPEN_PARENTHESIS hyper_exp CLOSE_PARENTHESIS cycle1

cycle1 ->

- OPEN_KEY statement_loop CLOSE_KEY

read ->

- READ OPEN_PARENTHESIS variable_loop CLOSE_PARENTHESIS SEMICOLON

variable_loop ->

- variable variable_loop1

variable_loop1 ->

- COMMA variable variable_loop1
- epsilon

write ->

- WRITE OPEN_PARENTHESIS hyper_exp_loop CLOSE_PARENTHESIS SEMICOLON

hyper_exp_loop ->

- hyper_exp hyper_exp_loop1

hyper_exp_loop1 ->

- COMMA hyper_exp hyper_exp_loop1
- epsilon

function_call ->

- ID function_call1 OPEN_PARENTHESIS function_call2 CLOSE_PARENTHESIS SEMICOLON

function_call1 ->

- DOT ID
- epsilon

function_call2 ->

- hyper_exp_loop
- epsilon

class_function ->

- AT_CLASS ID FUNCTION ID OPEN_PARENTHESIS parameter CLOSE_PARENTHESIS RETURNS return_arg OPEN_KEY function_statement_loop function_return CLOSE_KEY

function_declaration ->

- FUNCTION ID OPEN_PARENTHESIS parameter CLOSE_PARENTHESIS RETURNS return_arg OPEN_KEY function_statement_loop function_return CLOSE_KEY

function_return ->

- RETURN hyper_exp SEMICOLON






- epsilon



function_statement_loop ->

- statement_loop
- epsilon



Control de cambios








Retroalimentación para Avance #0

Cambio propuesto	Comentario	Resuelto
<ul style="list-style-type: none"> • creo que olvidaron el statement en el ClassFunction Declaration • Me perdí.. El Class Function Declaration es solo los Headers?..porque luego hay un Class Function..¿? 	Sí, uno es el puro header dentro del cuerpo de la declaración de la clase, y externo a éste se define el cuerpo con statements.	
<ul style="list-style-type: none"> • Sugiero poner SOLO UNA Class declaration donde la herencia sea opcional 	Ya lo cambiamos a todo manejarlo en sólo Class con herencia opcional, también la renombramos a class declaration.	
<ul style="list-style-type: none"> • Los arreglos se deben indexar con EXP no con id (se declaran con Ctelnt) pero se usan como A[j+1].... 	Creamos variable para agrupar id, acceso de elemento de arreglo y atributo de clase. Indexamos arreglo con EXP. En assignment, cambiamos de id's a variable. Agregamos variable a Factor.	
<ul style="list-style-type: none"> • El Write es de EXP no de ids 	Listo	
<ul style="list-style-type: none"> • Se asigna y se lee en una VARIABLE que no necesariamente es un id simple (puede llevar dimensiones ej: A[j+1]) 	Listo	

o incluso atributos miCarro.llantas		
<ul style="list-style-type: none"> En la llamada NO es Parameter (eso es en la declaración) los argumentos son EXP ej: Factorial (j*3) y..¿van a aceptar anidamiento de clases?..o para qué pusieron el id.id con ciclo en las llamadas?.. 	Cambio en function call de PARAMS a Hyper Expressions. Borramos el ciclo en las llamadas de función a herencia simple	
<ul style="list-style-type: none"> En Factor el id NO es simple necesariamente.. puede ser una casilla de un arreglo.. o incluso un atributo.. por eso les conviene definir una regla de VARIABLE.. revisen el archivo de Diagramas Comunes que está en el drive..les puede ser útil.. 	Borramos id de factor y dejamos la pura VARIABLE que agregamos previamente.	

Cambios durante el desarrollo de Avance #1

Cambio propuesto	Comentario	Resuelto
Necesitamos la distinción entre la palabra reservada int y el regex que matchee con integer value, lo mismo para float, string, bool.	Agregados ambos tokens. Cambiamos también el diagrama.	
Declarar sólo una variable por instrucción. Quitarlo de draw.io	Implementado y diagrama corregido.	

Agregar palabra reservada obj, int_value, float_value.	Agregadas en lista de palabras reservadas.	
CAMBIAR function parametros, separarlos con comma	Agregados parámetros como opcionales y poder agregar múltiples parámetros separados por comas.	
Corregir diagrama de class function para agregar flecha que evite la necesidad de tener un return	Agregada.	
Agregar bool_value y string_value a factor. E.g. "return true;"	Agregados.	
Cambiar parámetros de creación de objeto a una lista de hyper expressions opcionales.	Agregado.	
Agregar epsilon al diagrama de parameter.	Agregado	
Hacer cambio en la estructura de class function (mover @class al inicio) similar a Persona::saludar() en c++, "parche guadalupano" para diferenciar entre funciones	Agregado	

Cambios para considerar después de Avance #1

Cambio propuesto	Comentario	Resuelto
El return podría ser un statement, ya que no necesariamente va al final. E.g <pre>if (a > b) { return a; } else {</pre>	Podemos considerarlo como stretch goal.	

<pre>return b; }</pre>		
Agregar >=, <=	Podemos considerarlo como stretch goal.	